The Fuzzy Feeling SAS Provides

Electronic Matching of Records

without Common Keys

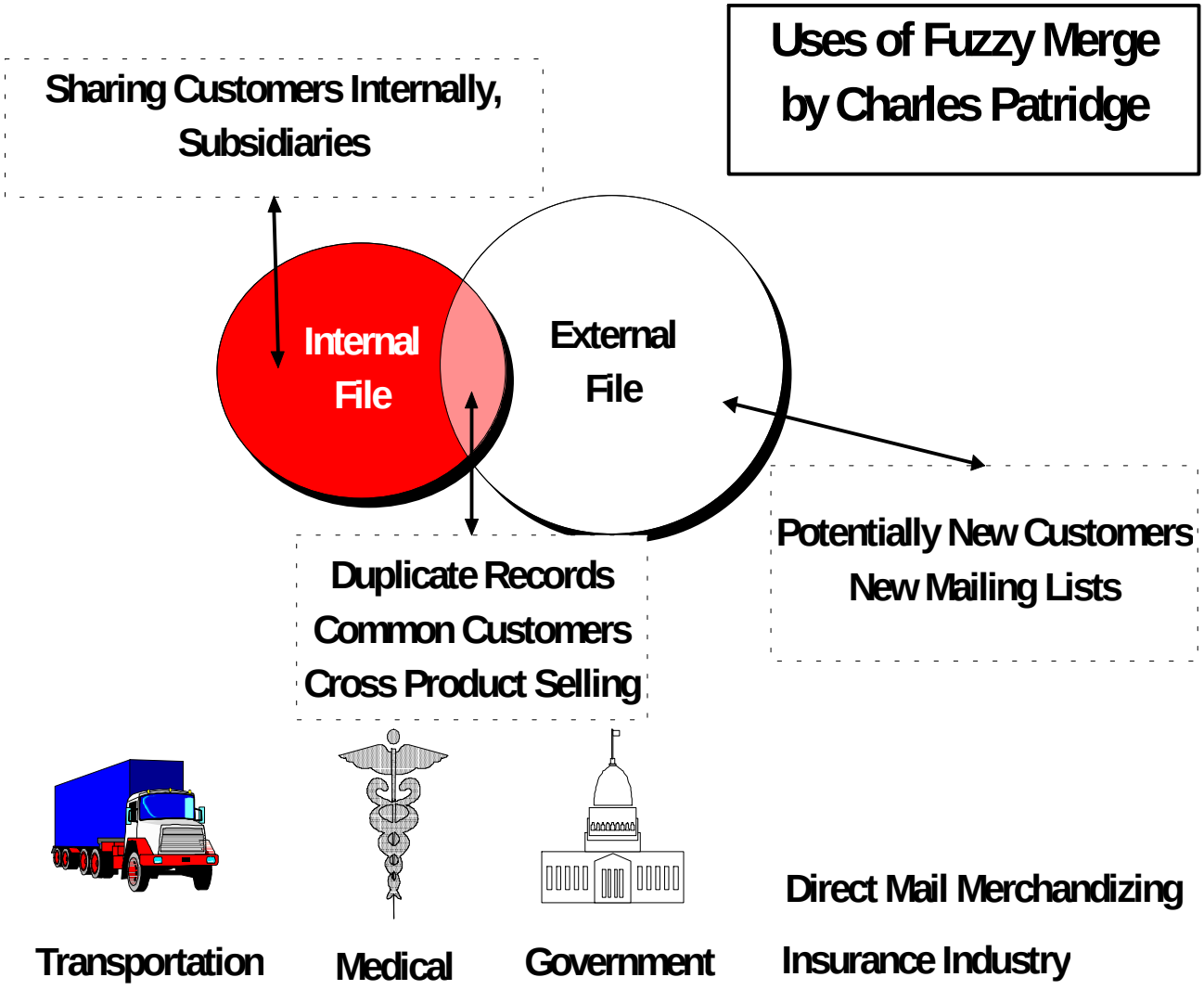
by

Charles Patridge
ITT Hartford Insurance
Corporate Actuarial
Hartford Plaza
Hartford, CT 06115
860-547-6644

SUGI 22
March 16-19, 1997
San Diego, CA

Contact Information:

Charles Patridge
172 Monce Road
Burlington, CT 06013
Home: 860-673-9278
Email: TVJB41A@prodigy.com
Website: htttp://pages.prodigy.com/SASCONSIG/saconsig.htm


Uses of Fuzzy Merge
by Charles Patridge
Sharing Customers Internally,
Subsidiaries
Internal
File
External
File
Potentially New Customers
New Mailing Lists
Duplicate Records
Common Customers
Cross Product Selling
Transportation
Medical
Government
Direct Mail Merchandizing
Insurance Industry

Introduction:

We begin by briefly describing two real life situations where this application has been implemented. I hope to provide a better understanding of when and where Electronic Fuzzy Matching could be effective and utilized. The remainder of the paper discusses the techniques and examples of the data and the SAS code attached.

Application A: The Mailing Lists

The Vice President of Marketing comes down with a list of clients from Dunn & Brad Street that are potentially ideal candidates for a special marketing product campaign. The VP asks you which ones (from the list) are and are not current customers. You ask if this list of clients exist on a floppy diskette, as you can see this list encompasses several hundred entries. The VP says "Sure, here is the diskette and this piece of paper describes what, where and how the fields are contained on the diskette".

You are relieved you do not have to key in these several hundred entries. You put the diskette into your "hot little" PC and start to peruse the file. You notice several things about the way the data is stored on this PC file:

1. There is no common "KEY" in this PC file that can be used to link these entries with your company's customer file directly, such as a social security number.

2. But this file has name, address, city, state and zip code for each entry.

You conclude that a SAS program could read your customer base file and the D&B file, sort the name of client by zip code and use a MERGE statement matching both files by zip code and name of client. After developing the initial SAS program and running your merge statement, you discover you have SUCCESSFULLY matched 21% of the D&B file with the company customer file. You deliver your report to the VP and state you have performed magic with the D&B file by matching 21% of these records. The VP states he appreciates your efforts but believes there should be more than 50% of these D&B entries in the current company customer file. You ask "How, can that be? I matched those records with a SAS program based on name of client with in zip code!". The VP says "I had my secretary take the first 100 entries and print them out, and she cross referenced them manually with our customer list. You then asked to see her list and those she matched. You review the secretary's list quickly and discovered the name of clients on the D&B list are spelled or positioned differently than on the company's customer list. For example:

D&B list	**Company Customer List**
Chuck Patridge P D P C, Ltd. 172 Monce Road Burlington, Ct. 06013	Patridge, Charles PDPC, Ltd. Monce Rd. Burlington, Ct. 06013-2545

You realize by looking at the two similar entries, they are in fact the same client. Yet, your SAS program with the MERGE statement has failed to match these records, because the name of the client is stored or spelled differently from your internal company customer file versus the D&B external file. With a puzzled looked, you reply to the VP "I can't match these two files in this way because D&B does not store their clients' names the same way our company does! We would need to purchase a specialized matching software package to perform your request". The VP says "How much would that cost?". You reply "About $60,000?". VP asks "Can't we develop our own matching program?" You reply "Sure, but I would need 3 or 4 months to develop such a program. But I will not be able to get to it for another 3 months unless you say it is OK to put the current projects on the back burner". The VP answers by saying "No, continue with the current projects. I'll have my secretary match the lists manually".

Application B: Keeping Track of Insurance Agents

Company A is a direct and reinsurance company, and its clients are other primary insurance companies. Each month each primary insurance company sends Company A list of insureds, as well as the insurance agent who placed the business. Company A also sells its own products to these same insurance agents. Company A would like to know which agents of

the primary companies are also their own agents and which are not. Company A's agent list has more than 55,000 members (agents). Each month of submissions to Company A could easily be one to two thousand from the primary companies. And as before, each primary company has its own way of storing mailing data on its agency force.

Question at hand, how many agents of the primary companies are agents of Company A's ?

I chose to start this presentation with a couple real life business situations I have encountered over the past 16 years, although the names have been disguised to protect the innocent. I have used this particular program for various applications similar to the above examples with only minor changes to the actual code.

The main purpose of this application is the ability to match records from two separate files which do not have any common key to link records together. Instead, the files have common fields but can not be matched exactly by the contents of these fields due to the way the data is keyed or maintained. For instance, the client name in file A is keyed as “Charles Patridge”; in file B the client name is “Chuck Patridge”. A computer can not conclude that these two data fields are, in fact, the same person. However, an individual can create a computer program(s) to consider these two data fields to be close to being a match. And that is what I have attempted to create.

As with most applications, the critical aspect to this particular problem was becoming familiar with the data. To accomplish this task, I took approximately 1,200 records from a real file and started to match each record with a file consisting of more than 55,000 similar type records. As I matched each record, I analyzed the data from both files and eventually determined an algorithm could be created to simulate what the human brain does when comparing two similar character strings.

To duplicate this human thought process and utilize the current tools built within the base SAS system, I began the initial program one module at a time.

First, I decided the SCAN function would be crucial to the success of my program. In using SCAN, I did not have to worry about the order in which a client's name is stored. That is, Charles Patridge is the same client name as Patridge, Charles. All that is important is the "words" in one field are "contained" in a character data string of another field. In addition, I realized not all words from one field should or have to be contained in another field. That is, "Charles Patridge" is the same person as "Patridge, Chuck". The difference between these two comparisons is "Chuck" and "Charles".

To compensate for these nicknames and/or abbreviations, one can build a SAS macro to "normalize" the data being the second and most important ingredient. For instance, when dealing with addresses, there are certain words which are commonly abbreviated: St for Street, Rd for Road, Dr for Drive, etc. Hence, I had to build several SAS macros to "normalize" my data depending on the field's use. For example, addresses need one kind of normalizing versus the names of clients. A special note is needed when trying to normalize the data. I have found data contents have a certain characteristic based on industry and/or application. One example is the insurance agent application described above. There are numerous insurance agents throughout the United States whose business/corporate name contain the word "AGENCY". However, in the DMMC example above, "AGENCY" can be a rather unique word and be useful in the matching process. The point is "normalizing" data should be done with care for each application to maximize the matching (hit ratio) process.

The last ingredient needed for my application to work is the ability to read one record from one SAS dataset and hold it open while reading through another SAS dataset to look for potential matches with the record from the first file. This third ingredient was accomplish using the POINT-- option of the SET statement. A side benefit of the POINT option was the ability to reduce the amount of I/O and CPU by breaking down the source file (file to be used to determine if a match exists) into subsets based on the application. For instance, the example Application B tries to match insurance agents from all across the United States. I decided to subset the source file into 50 datasets based on the state in which the agent's address is located. This allow the program to search records based on state instead of searching through all states for a possible match.

To enhance the matching process and reduce the manual effort in matching records, I developed several additional techniques. First, I

knew I needed to prioritized those matches found. That is, there could be numerous matches in a given situation, and I wanted the most probable match to occur first by some determined method. The method I chose was to count the number of words found in each possible occurrence, and then rank these matches by the number of words contained in descending order. In real life applications, there could be numerous matches (especially in Metro areas) for a given client, and I did not want the computer program to select just one match. I needed to list all possible matches for a person to make the final decision as to which one(s) are truly the real matches. This is an application where the computer can do alot of the grunt work but is limited in making the right/accurate choice. Hence, I needed to sort the final matches according to the source record with the most words found.

Another technique added to the matching process was the occurrence of vowels within the data. Due to human error, I determined many words are misspelled because of vowels. For example, "Charlie" sounds the same as "Charley" but are spelled differently because of the vowels. So, to enhance the matching process, I match each "normalized" word first looking for a possible hit. And then I squeeze out the vowels and perform the same routine looking for a hit by using only the consonants. For example, "Charlie" is not contained in the string "Charley". However, after removing the vowels, "Chrl" is contained in "Chrl". And I just have increased my hit ratio by one word.

A side note for SAS users of release 6.07, I tried using the SOUND() function in my application. It would have required to parse each and every word in the source file adding a significant amount of processing to the matching process. The gains I received did not seem to warrant utilizing it. However, it could be used for those cases where it seems to make sense such as "BOOK" vs "HOOK", "COUCH" vs "POUCH", etc. In addition, SOUNDEX function can be extremely useful but also has a tendency to generate excessive “false” hits. I have placed a macro on the website (listed at the end of this paper) which uses the SOUNDEX function for matching purposes and it much more elegantly simplified in performing non-keyed matching but can generate numerous false hits.

Having discussed the basic concepts and necessary ingredients to the application, it is time to discuss the actual program using the example Application B. First, it is assumed both files (source-master and target-to be matched) have been "normalized" and edited for obvious errors (state code matches zip code, all characters are upper case, etc.).

This application involves several hundred to several thousand records each month submitted to Company A from approximately 30 primary insurance companies. The media for submission is either on paper (keyed) or on a magnetic tape/diskette. The required data fields are agent number, agent name, address, city, state, zip code, and company code. It should be noted there are some intervening programs needed to separate certain fields due to the way companies submit their data. For instance, many companies code city and state as one field, and Company A has program routines to separate this into two separate fields. However, my major program does not really need the state or city, as I use the zip code as the primary key to determine if a possible match record is within a postal boundary territory. More specifically, I use only the first three digits of the zip code to determine if one record is within a geographic area to search for a possible match. Once the files are in order, the process of matching the new monthly agent records against Company A's 55,000 agent master list begins. First, determine if a record has already been matched for a given company's agent. If so, then bypass matching it. If not, store record in a separate SAS dataset for matching. Read the historical matches to see if any prior records have not been matched and recycle them for another round of matching.

Start the physical process of matching. If a possible match is found then store the target and source record in another SAS dataset. If no match is found then mark target record as no match. Due to the way the data is entered into Company A's target files, several attempts of matching are needed:

Match target name 1 field to source name 1 field.

Match target name 2 field to source name 1 field. *

Match target name 2 field to source addr 1 field. *

Match target addr 1 field to source addr 1 field.

* name 2 field may contain either name of client or 1st address line of client (hence, need to match against name and addr).

After all target records have been through the matching process, the matched records are then accumulated and sorted by the most probable match first. These records are then printed and save to a SAS dataset for an on-line selection system. Here, the operator using a SAS/FSP application can select which records are truly a match. The on-line system is an added component to the base application due to the need to have SAS/FSP. Otherwise, the printed version can be used to visually select which records are indeed matched. Once manually marked, an operator can use the system file editor and go through and mark/update the source file with the appropriate matches. Once source files are updated, the process can start all over again for the next month.

In a typical month of about 1,500 records, I am able to run the entire application from start to finish (includes selecting match records and updating source file) in approximately 3 to 4 hours. Compare this to manually matching 1200 records (using system file editor) and taking approximately 8-10 hours with a rather responsive IBM 3090 VM/CMS system, the application made me significantly more productive. And in addition, the application ended up being more accurate than manually doing it due to human factors.

In order to keep this paper brief and within SUGI guidelines, I have intentionally left out all macros, "normalizing" routines which do not directly support the matching process and the matching programs as they can be downloaded via the internet as given at the end of this paper.

I hope this application and the other examples listed above have demonstrated the usefulness and flexibility of the SAS system, as well as the fuzzy feeling SAS can provide! I thank you very much for your time and desire to hear about Fuzzy merges. I would be delighted to hear from you on how this routine has been used within your environment and the reults. I can be reached for consultation on the software routine at the address below, and the software can be downloaded via the Internet listed below.

Charles Patridge
172 Monce Road
Burlington, CT 06013
Home: 860-673-9278
Email: TVJB41A@prodigy.com

Website: http://
pages.prodigy.com/SASCONSIG/saconsig.htm
(type exactly as seen, the name is Case Sensitive)

Since I am the author and owner of this code, I am explicitly providing access and duplicating privileges to these matching SAS programs via the internet, as long as ownership is duly published and recognized.

Select Tips and Techniques and look for Fuzzy merge narrative and then download using your Web browser**.**